(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 772 836 B1

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention of the grant of the patent:
12.12.2001 Bulletin 2001/50

(21) Application number: **95920913.1**

(22) Date of filing: **05.06.1995**

(51) Int Cl.[7]: **G06F 17/30**

(86) International application number:
PCT/FI95/00319

(87) International publication number:
WO 95/34155 (14.12.1995 Gazette 1995/53)

(54) A METHOD FOR STORING AND RETRIEVING DATA AND A MEMORY ARRANGEMENT

VERFAHREN ZUM DATEN SPEICHERN UND DATEN WIEDERFINDEN UND EINE SPEICHERANORDNUNG

PROCEDE POUR ENREGISTRER ET EXTRAIRE DES DONNEES, ET SYSTEME DE MEMOIRE

(84) Designated Contracting States:
AT BE DE FR GB IT NL SE

(30) Priority: **06.06.1994 FI 942663**
06.06.1994 FI 942664

(43) Date of publication of application:
14.05.1997 Bulletin 1997/20

(73) Proprietor: **Nokia Networks Oy**
02150 Espoo (FI)

(72) Inventor: **TIKKANEN, Matti**
FIN-02710 Espoo (FI)

(74) Representative: **Äkräs, Tapio Juhani et al**
Oy Kolster Ab,
Iso Roobertinkatu 23,
P.O. Box 148
00121 Helsinki (FI)

(56) References cited:
EP-A- 0 650 131 **US-A- 5 319 777**

- **IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Volume SE-5, No. 4, July 1979, JON L. BENTLEY, "Multidimensional Binary Search Trees in Database Applications", pages 333-340.**

Note: Within nine months from the publication of the mention of the grant of the European patent, any person may give notice to the European Patent Office of opposition to the European patent granted. Notice of opposition shall be filed in a written reasoned statement. It shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 0 772 836 B1

Description

[0001] The invention relates to methods in accordance with the preambles of the appended claims 1 and 10 for storing data, and to methods in accordance with the preambles of the appended claims 6 and 12 for retrieving data. The invention also relates to associative memory arrangements in accordance with the preambles of the appended Claim 14.

[0002] The approach in accordance with the invention is intended for use primarily in connection with central memory databases. A suitable application is the maintenance of a subscriber database in a telephone exchange.

[0003] The prior art unidimensional directory structure termed digital trie (the word "trie" is derived from the English word "retrieval") is the underlying basis of the principle of the present invention. The digital trie is a tree-like structure composed of two types of nodes: leaf nodes containing a record, and internal nodes guiding the retrieval. An internal node is an array having a size of two by the power of k (2^k) elements. If an element is in use, it refers either to an internal node at the next level in the directory tree or to a leaf node containing a record. In other cases, the element is free (empty). Search in the database proceeds by examining the search key (which in the case of a subscriber database in a telephone exchange, for instance, is typically the binary numeral corresponding to the telephone number of the subscriber) k bits at a time. The bits to be searched are selected in such a way that at the root level of the structure (in the first internal node), k leftmost bits are searched; at the second level of the structure, k bits next to the leftmost bits are searched, etc. The bits to be searched are interpreted as an unsigned binary integer that is employed directly to index the element array (the index indicates a given element in the array). If the element indicated by the index is free, the search will terminate as unsuccessful. If the element refers to an internal node at the next level, k next bits extracted from the search key are searched at that level in the manner described above. As a result of comparison, the routine branches off either to an internal node at the next level or to a leaf node typically containing a key-pointer pair. If the element refers to a leaf node containing a record, the key stored therein is compared with the search key. The entire search key is thus compared only after the search has found a leaf node. Where the keys are equal, the search is successful, and the desired data unit is obtained at the storage address indicated by the pointer of the leaf node. Where the keys differ, the search terminates as unsuccessful.

[0004] Figure 1 illustrates an example of a digital trie structure in which the key has a length of 4 bits and k=2, and thus each internal node has 2^2=4 elements, and two bits extracted from the key are searched at each level. Leaf nodes containing a record are denoted with references A, B, C, D...H...M, N, O and P. Thus a leaf node is a node that does not point to a lower level in the tree. The internal nodes are denoted with references V1...V5 and internal node elements with reference VA in Figure 1.

[0005] In the exemplary case shown in Figure 1, the search keys for the leaf nodes shown are as follows: A=0000, B=0001, C=0010,...,H=0111, ... and P=1111. In this case, a pointer is stored in each leaf node to that storage location in the database SD at which the actual data, e.g. the telephone number of the pertinent subscriber and other information relating to that subscriber, is to be found. The actual subscriber data may be stored in the database for instance as a sequential file of the type shown in the figure. The search is performed on the basis of the search key of record H, for example, by first extracting from the search key the two leftmost bits (01) and interpreting them, which delivers the second element of node V1, containing a pointer to node V3 at the next level. At this level, the two next bits (11) are extracted from the search key, thus yielding the fourth element of that node, pointing to record H.

[0006] Instead of a pointer, the leaf node may contain (besides a search key) an actual data file. Thus for example the data relating to subscriber A (Figure 1) may be located in leaf node A, the data relating to subscriber B in leaf node B, etc. Thus in the first embodiment (in which a key-pointer pair is stored in the leaf node), the digital trie structure serves as a directory structure only, and in the second embodiment (in which the key and the actual data are stored in the leaf node) it serves as an associative memory. (In an associative memory, the storage locations are identified on the basis of their contents.)

[0007] In the approach of the present invention, the search key is multidimensional in contrast to the above, that is, the search key comprises a number of attributes (for example the family name and one or more surnames of a subscriber), and thus k_i search key bits of an attribute are searched at a time in each attribute (i is an index corresponding to an attribute).

[0008] Previously, the unidimensional trie structure described above has been expanded into a multidimensional "EXCELL" structure (abbreviation for the words "extendible cell"), which is designed to support searches relating to geometric applications for maximum efficiency. The EXCELL structure comprises cells partitioning the search space, and a separate directory part. The cell division is executed by recursive subdivision of the search space into two equal parts alternately in the direction of each axis of the search space. The result of the cell division is a multidimensional trie structure. The directory part is a grid in tabular form, wherein each grid element is equal to the smallest cell in the search space. The grid element points to a cell in the search space, and more than one grid elements may point to one cell. (The EXCELL structure is described in Tamminen, M.: *The EXCELL method for efficient geometric access to data*, Helsinki 1981, Acta Polytechnica Scandinavica, Mathematics and Computer Science Series #34, to which the

interested reader is referred for a more comprehensive exposition.)

[0009] Such a multidimensional trie structure, in which the cell size is dynamic (the directory part is doubled when data are entered into the structure), is attended by the problem of fast (exponential) growth of the directory part caused by certain kinds of (non-uniform) search key distribution. In consequence, the storage occupancy (memory circuit requirement) will increase inordinately and the efficiency (speed) of storage management will be impaired.

[0010] It is an object of the present invention to eliminate the problems described above and to provide a memory in which the directory part can be contained in a smaller storage space than hitherto and which will be easier to manage than previously. This object is achieved with the storage and retrieval methods of the invention, the storage method being characterized in its basic form by (a) selecting from the search key related to each dimension a predetermined dimension-specific number (k_i) of bits and forming therefrom a search word on the basis of which the address of the next node is sought from an internal node at the root level of the tree-shaped hierarchy, and proceeding to said node, (b) selecting from the unselected bits in the search key related to each dimension a predetermined dimension-specific number (k_i) of bits and forming therefrom a search word with which the address of a further new node at a lower level is sought from the array of the node that has been accessed, (c) repeating step (b) until an empty element has been encountered or until the address of the new node at a lower level is the address of a leaf node, and (d) storing a pointer in the leaf node and said data unit at the storage location indicated by the pointer or alternatively storing said data unit in the leaf node. The retrieval methods in accordance with the invention are, in their basic form, characterized by that which is set forth in the characterizing portion of the appended claim 6. For the basic forms referred to above, a special case in which the directory structure comprises one node only may also be employed. The storage methods in accordance with this embodiment are characterized in that which is set forth in the characterizing portion of the appended claim 10, and the retrieval methods are in turn characterized in that which is set forth in the characterizing portion of the appended claim 12. The associative memory arrangements of the invention are in turn characterized in that which is set forth in the characterizing portion of the appended claim 14.

[0011] The idea of the invention is to perform address computation in a multidimensional trie structure in such a way that a given predetermined number of bits is selected from each dimension independently of the other dimensions, a typical implementation being such that in order to calculate one element array index, an unequal number of bits is selected from the search keys of the different dimensions. Hence a fixed limit independent of the other dimensions is set for each dimension in an individual internal node in the trie structure, by predetermining the number of search key bits to be searched in each dimension. Thus the selection in each dimension is independent of the selections in the other dimensions. This course of action avoids excessive growth of the structure in the lateral direction on the one hand (the size of the internal node has an upper limit), and excessive growth of the structure in depth in the direction of any of the dimensions on the other hand (depth, i.e. the number of levels in each dimension, is equal to the total length of the search key in bits divided by the number of bits to be searched at a time, i.e., also the depth can be influenced by a parameter determining the size of the internal node). Since the directory can thus be given specific limits in the lateral and depth directions in advance, storage requirement will also be curbed.

[0012] With the approach of the invention, the need for memory circuits in the hardware will be smaller than heretofore, and the storage management will be effective, particularly when the distribution of the values of the search keys is known in advance (which is typically the case in a telephone exchange, for instance).

[0013] Another advantage of the invention is that it is possible to design the structure (by selecting the parameters) so as to favour, besides full key retrievals, also partial key retrievals equitably in all directions. The method also permits retrievals in the directions of certain dimensions only, i.e. partial key retrievals, with maximum efficiency. These characteristics are of significance particularly in real-time databases requiring high speed. A further advantage in terms of speed is that the approach enables efficient parallel processing (in other words, if for instance the search branches off into two subtrees, each can be processed with a separate processor).

[0014] In the following the invention and its preferred embodiments will be explained with reference to Figures 2 to 6 by way of examples in accordance with the accompanying drawings, in which

Figure 1 illustrates the use of an unidimensional digital trie structure in the maintenance of subscriber data in a telephone exchange,
Figure 2 illustrates a multidimensional trie structure,
Figure 3 illustrates the structure of an element in an internal node in the trie structure in accordance with the invention,
Figure 4 illustrates nodes in the directory structure employed in the memory in accordance with the invention,
Figure 5 illustrates ways in which an element in an internal node (an element array index) can be determined, the internal nodes being in accordance with Figure 4, and
Figure 6 illustrates a memory arrangement in accordance with the invention on block diagram level.

[0015] As stated previously, in the present invention the trie structure has a multidimensional implementation (gen-

erally n-dimensional). Such a multidimensional structure is otherwise fully similar to the unidimensional structure described at the beginning, but the element array comprised in the internal node is multidimensional. Figure 2 exemplifies a two-dimensional $2^{2*}2^{1}$ structure, in which one dimension in the element array comprises four elements and the other dimension two elements. Leaf nodes pointed to from the elements in the internal node are indicated with circles in the figure.

[0016] Address computation in the multidimensional case is performed on the same principle as in the unidimensional case. The fundamental difference, however, resides in that instead of one element array index, an index is calculated for each dimension in the element array (n indices). Each dimension thus has a search key space of its own {0, 1,..., 2^{v_i}-1} (v_i is the length of the search key in bits in each dimension and $i \in \{1,...n\}$).

[0017] In order to represent the configuration of an individual element VA in an internal node, it is first defined what is understood under the term overflow element. When a new pointer is added to the directory tree (as a result of data inserted into the memory), it may be at such a location in the internal node that already has a pointer referring to a leaf node. Such a situation is termed a collision. There are a wide variety of methods for resolving collision situations. In view of curbing memory requirement, it is no use creating a new internal node for the colliding pointer at the next level of the directory tree, but the pointer is located in the element of the colliding internal node if that node contains free elements. Such elements are called overflow elements, and they thus comprise pointers that are not located in accordance with the above straightforward address computation mechanism. In order that such records may be retrieved, the elements in the internal node must incorporate a separate identifier indicating the type of the pointer contained in said element, in other words, whether it refers to an internal node at the next level or to an overflow element (in the same internal node).

[0018] Figure 3 exemplifies the structure of an element in the internal node in the directory structure of the memory in accordance with the invention. The internal node may be of the type directory node or overflow node. The type of the internal node is directory node if it contains no overflow elements, and overflow node if it contains even one overflow element. An element in turn may be a directory element, an overflow element, or a free element (not in use). A directory element can refer to a directory node, overflow node, or leaf node. An overflow element, on the other hand, can only refer to an overflow node or to a leaf node.

[0019] Each element in this example has a length of 40 bits (5 bytes), comprising the following fields (length of field in parentheses): pointer field 31 (32 bits), pointer type field 32 (2 bits), and element type field 33 (2 bits). Furthermore, one vacant field 34 is left in the element. The type of the pointer can be encoded with the two bits of the pointer type field 32 for example as follows:

- 00: pointer not in use,
- 01: pointer to a directory node,
- 10: pointer to an overflow node, and
- 11: pointer to a leaf node.

[0020] The type of the element can be encoded with the two bits of the element type field for example as follows:

- 00: free element,
- 01: directory element,
- 10: overflow element, and
- 11: not in use.

[0021] In the multidimensional trie structure of the memory in accordance with the present invention, the size of each internal node (number of elements) is predetermined by fixing the parameter k_1 of each dimension (number of search key bits to be searched in the dimension) in advance on the basis of the desired size of an individual internal node in the direction of that dimension, and on the other hand on the desired depth of the structure in the direction of that dimension. The size of the internal node in the direction of each dimension is thus 2^{k_i} elements, and the total number S of elements in the internal node is also a power of two:

$$S = \Pi\, 2^{k_i} = 2^{k_1} \times 2^{k_2} \times 2^{k_3} \times \ldots = 2^{N} \qquad (1)$$

[0022] All elements in an internal node having n dimensions can thus be pointed to with n integers ($n \geq 2$), each of which can have a value between $\{0,1...2^{k_i}-1\}$. Thus the predetermined parameters include the total length of the search key in each dimension and the number of bits searched in each dimension. If for example one dimension of a search key has 256 attributes (e.g. first names) at the most, the total length of the search key is 8 bits. If the number of bits

to be searched at one time is selected to be for instance 1, the structure has a depth of 8/1=8 in the direction of this dimension. By suitably selecting the number of searched bits in each dimension, for example depth can be set to be equal in all dimensions.

[0023] Hence in accordance with the invention the number of searched bits can differ in each dimension, and thus also the need for storage space can be controlled individually for each dimension.

[0024] Figure 4 shows an example of an internal node V10 used in the directory structure of the memory in accordance with the invention, employing a three-dimensional search key. In the direction of the first dimension (x), the internal node has 2^2=4 elements, in the direction of the second dimension (y) it has 2^1=2 elements, and in the direction of the third dimension (z) it has 2^3=8 elements, which gives a total of 2^6=64 elements numbered 0...63.

[0025] Since the memory space in practical hardware implementations (for example computer equipment) is unidimensional, the multidimensional array is linearized, i.e. converted to be unidirectional, in the address computation operation (that is, in proceeding in the search tree). The linearization is an arithmetic operation that can be performed on arrays of all sizes.

[0026] In linearization, the elements in the array are numbered starting from zero (as shown in Figure 4), the number of the last element being one less than the product of the sizes of all dimensions. The number of an element is the sum of the products of each coordinate (for example in the three-dimensional case, the x, y and z coordinates) and the sizes of the dimensions preceding it. The number thus computed is employed directly as an index for the unidimensional array.

[0027] In the case of the internal node shown in Figure 4, the element number VA_n is calculated in accordance with the above with the formula:

$$VA_n = x+y\times 4+z\times 4\times 2 \qquad (2)$$

where $x\in\{0,1,2,3\}$, $y\in\{0,1\}$ and $z\in\{0,1,2,3,4,5,6,7\}$. Thus for example for element 54 we obtain from the coordinates thereof (2,1,6): $2+1\times 4+6\times 4\times 2=2+4+48=54$.

[0028] When the (n-dimensioned) element array of an internal node of an n-dimensional trie structure is linearized, in accordance with the above the size of each dimension is 2^{k_i}, where k_i is the number of bits to be searched at a time in the dimension concerned. If a coordinate in accordance with the dimension is denoted by reference a_j ($j\in\{0,1,2,\ldots n\}$), the linearization can be written out as

$$\sum_{j=1}^{n} a_j \prod_{i=0}^{j-1} 2^{k_i} \;,\; \forall\, j: a_j \in \{0,1,\ldots,2^{k_j}-1\} \wedge k_0 = 0 \qquad (3)$$

[0029] The linearization can be carried out by performing a multiplication in accordance with formula (3); yet it is expedient to perform the linearization by forming from the search key bits a bit string, the corresponding numeral indicating the element whose content provides the basis for proceeding in the directory tree. In forming the bit string, the bits extracted from the search key of each dimension are set in succession, and the final bit string thus obtained is interpreted as a binary numeral. This gives the same result as formula (3).

[0030] Figure 5 illustrates the linearization to be performed both by way of multiplication and by way of bit string formation in connection with the internal node shown in Figure 4. As indicated previously, the numbers of search key bits to be interpreted in each dimension have been selected as follows: k_1=2, k_2=1 and k_3=3. The search keys in the direction of each dimension are as follows: a_1=1001, a_2=10, and a_3=110001. In forming the bit string, the two leftmost bits (10) of the search key a_1 in the first dimension (the first dimension is the one in the direction of which the numbering of the elements is commenced, starting from zero) are first extracted and taken as the least significant bits of the bit string to be formed. This is denoted by line 51. Next, the leftmost bit is extracted from the search key of the second dimension (line 52), and this bit is appended on the side of the most significant bits (left-hand side) into the bit string to be formed. Lastly, the three leftmost bits are extracted (110, line 53) from the search key of the third dimension, and these bits are inserted as the leftmost bits into the bit string. The resulting bit string (110110) is interpreted as a binary numeral, which will give access to element 54. The same element will be accessed also by way of multiplication in accordance with formula (3), but in that case the selected search key bits are interpreted first as a binary numeral, and coefficients for formula (3) are formed from the number thus obtained as shown in Figure 5. On the basis of the pointer contained in element 54, the next internal node is accessed, wherein the same operation is repeated with the unused

search key bits. (In the exemplary case, element 9=$(001001)_2$ is selected from this internal node.) Each element in the internal node may refer either to another internal node (as element 3 to internal node V11) or to a leaf node containing a record (as element 43 to record R). In other cases, the element is not in use (is empty).

[0031] With bit string formation the multiplication operations can be converted to addition operations and bit shifts (or to additions and bit-related OR operations). The mutual multiplication operations on numbers 2^{k_i} are replaced by addition operations on exponents k_i. The remaining multiplication of coordinate a_j by a power of two will be converted to a bit shift operation by bit string formation (a multiplication by 2^k is known to correspond to a bit shift of k bits to the left). As multiplication will always take at least 13 clock cycles (with an Intel i486 processor, for instance), a bit shift can be performed in three clock cycles when the operands are stored in the processor register, in four clock cycles when the operands are to be found in the cache of the processor, and in ten clock cycles even in the case that the operands are to be retrieved from memory. Addition, on the other hand, will only take one clock cycle in the registers and otherwise 2-3 or 8-9 clock cycles depending on whether the operands are to be found in the cache of the processor.

[0032] The above-described method by which a bit string is formed is advantageous in that it is order-preserving (in other words, the records are organized in the structure in the order of the values of their search keys). This is due to the fact that k_i most significant (leftmost) bits are extracted from the search key at a time. However, in principle it is possible to form the bit string also in such a way that k_i least significant (extreme right) bits are extracted from the search key.

[0033] The bit string may also be formed by treating the dimensions in a different order than above, yet the above method in which the bit string to be formed is increased on the left is the simplest, because in that case there is no need to know how many positions are to be assigned for the bits of each dimension in the bit string.

[0034] The multiplication in accordance with formula (3) may be employed for linearization (i.e., calculating an element array index) irrespective of the size of each dimension. On the other hand, bit string formation can only be employed where the size of the internal node (number of elements) in the direction of each dimension is a power of two. Since the internal node of the directory structure in accordance with the present invention meets this requirement, it is advantageous to employ the bit string-formation method in the retrieval and storage methods in accordance with the invention, which is a more efficient (rapid) method than the multiplication in accordance with formula (3). The rapidity is due to the fact that bit string formation will convert multiplications to additions and bit shifts, which are faster to perform.

[0035] A directory structure as set forth above may be varied for example in such a way that a leaf node in the structure comprises more than one key-pointer pair. The difference then resides in that several search key comparisons instead of one comparison are performed in the leaf node.

[0036] Instead of a pointer, the record contained in the leaf node may comprise (besides the search key which is not indispensable) actual data. In this embodiment, the trie structure thus serves as an associative memory.

[0037] In principle, storage maintenance includes three different operations: retrieval, addition (storage), and deletion. When a record is to be added to actual data (subscriber data at a telephone exchange, for instance), the addition is made either directly to a leaf node or elsewhere in the memory. In the latter case, a pointer to the storage address at which the data record is stored is stored in the leaf node.

[0038] Hence the search proceeds in such a way that a predetermined dimension-specific number of bits (k_i) is selected from the search key related to each dimension, and a search word is formed from the bits in the above manner, on the basis of which the element array index of the internal node at the root level is found. The element corresponding to the index will deliver the address of the next node, which is thus accessed. In the next node, a predetermined dimension-specific number of bits (k_i) is selected from among the unselected bits in the search key related to each dimension, and a search word is formed from these bits, employed for finding an internal node element. The selection of search key bits is continued until all bits in the search key have been selected or the address of a node at the next lower level is the address of a leaf node. Thereafter, either (a) the pointer contained in the leaf node is read from the leaf node and the record is read from the location indicated by the pointer, or (b) the one or more records contained in the leaf node are read. If an empty element is encountered in proceeding in the directory structure, the search is terminated as unsuccessful (in other words, it is known that no data associated with the relevant key is stored in the memory).

[0039] Hence in accordance with the invention a predetermined dimension-specific number of bits (k_i) is selected from the unselected bits of the search key related to each dimension. In each dimension, the number to be selected is independent of the number selected in the other dimensions, and the number to be selected remains constant in the same dimension.

[0040] When data are inserted, the search proceeds in the directory structure on the basis of a search key in a similar manner until an empty element is encountered or until the address of a new node at a lower level is the address of a leaf node.

[0041] In the former case, a leaf node is created and the free element is set to point to this leaf node. Thereafter, the data are inserted either directly into the leaf node or elsewhere in the memory; in the last-mentioned case, a pointer to the storage address at which the added record was stored is stored in the leaf node.

[0042] In the latter case, a leaf node already existing, the situation depends on whether the leaf node contains any space for an insertion (in other words, the situation depends on the size of the leaf node). If for example only one record will fit into the leaf node (or the leaf node is otherwise full), the depth of the directory structure must be increased. In that case, the element at the point of collision is set to refer to a new node in which element array indices are calculated for the keys. Adding of a new node is repeated if the indices are still equal. Thus the addition of a new node is repeated so many times that the indices are unequal. This is accomplished by means of identifying search keys at the latest when all bits in the search key are employed, i.e. when the depth of the structure has reached its maximum. If there is space in the leaf node, the pointer or the actual data record is stored in the overflow element of the leaf node. If a pointer is stored in the overflow element, it will again point to the storage address at which the actual data record is stored. The overflow possibility may only be realized in a leaf node of the directory structure.

[0043] Deletion of data is performed in a manner known per se by finding, on the basis of the search key, the node containing a reference to a pointer or record to be deleted, as described above. If possible, the node is freed in order that storage space occupancy may be maintained at a minimum all the time. Freeing is possible for example where the node contains one pointer to a leaf node and no pointer to a node at a lower level.

[0044] The collision-prevention strategy to be chosen in each particular case usually has an effect not only on the insertion operations but also on how the retrieval and deletion operations are to be performed. Thus the details of the implementation of these operations can vary, even though in all of them progress is made by the address computation method of the present invention. Since, however, such a variation is not relevant to the actual inventive idea and since the variation will take forms known to those skilled in the art, it will not be more closely explained in this context.

[0045] Figure 6 shows a memory in accordance with the invention on block diagram level. Each dimension has a dedicated input register, and hence there are n input registers. The search key of each dimension is stored in these input registers, each key in a register of its own. The input registers are connected to register TR in which the above-described search word is formed. The register TR is connected via adder 61 to the address input of memory MEM. The output of the memory in turn is connected to address register AR the output of which in turn is connected to adder 61. Initially the bits selected from each register are read into the common register TR in the correct order. The initial address of the first internal node is stored in the address register AR, and the address obtained as an offset address from register TR is added to the initial address in adder 61. The resulting address is supplied to the address input of the memory MEM, and the data output of the memory provides the initial address of the next internal node, the address being written into the address register AR over the previous address stored therein. Thereafter the next selected bits are again loaded from the input registers into the common register TR in the correct order, and the array address thus obtained is added to the initial address of the relevant array (i.e., internal node), obtained from the address register AR. This address is supplied to the address input of the memory MEM, the data output of the memory thereafter providing the initial address of the next internal node. The above-described procedure is repeated until the desired position has been accessed and recordal can be performed or the desired register read.

[0046] The rapidity of the method of the invention can be influenced by the type of hardware configuration chosen. Since progress is by way of the above-stated bit manipulations, by selecting a given dimension-specific number of bits at a time from the search key of each dimension, address computation can be sped up by shifting from use of one processor to a multiprocessor environment in which parallel processing is carried out. An alternative to the multiprocessor environment is an ASIC circuit.

[0047] In the above, the structure of the memory has been described in the form that it will typically assume in implementation. In practice, nonetheless, one leaf node is first created from all nodes, and only after that first leaf node has overflown is the first internal node created, and the above-described directory structure will start taking shape. Thus in principle it is even possible to realize address computation of the above-described kind in one very large leaf node (that will never overflow) that comprises a multidimensional array, one element in the array thus containing - if it is in use - a data unit or a pointer to a data unit. Hence in this case only the first k_i bits are selected from the search key of each dimension. If the element found by means of these bits is empty, either an actual data record or a pointer to that storage address at which the data record is stored is stored in the element. If a collision situation arises in the storage, it will be resolved by methods known per se. Such a one-leaf embodiment may in practice be feasible if the distribution of stored data is appropriate. The efficiency of storage space utilization, however, is inferior to that of the basic embodiment disclosed above, which has a multidimensional directory structure.

[0048] The one-leaf additional embodiment described above is, in a sense, a special case of a possible embodiment which also has a multidimensional directory structure, but along with that, address computation is continued in the leaf node as well, as has been described previously (provided that unsearched bits still exist). In that case, a leaf node has a structure similar to that of an internal node.

[0049] Even though the invention has been explained in the above with reference to examples in accordance with the accompanying drawings, it is obvious that it is not to be so restricted, but it can be modified within the scope of the appended claims.

Claims

1. A method for storing data identifiable by a search key in memory, the data being stored as data units in a dedicated storage space assigned for each data unit, wherein storage is performed by proceeding, on the basis of a search key associated with the data unit to be stored, in a directory structure comprising a tree-shaped hierarchy having nodes at several different levels, wherein an individual node can be

   - an internal node comprising a multidimensional array wherein an individual element may contain the address of a lower node in the tree-shaped hierarchy and wherein an individual element may also be empty, or
   - a leaf node in which at least one data unit is stored or which contains at least one pointer to a stored data unit,

   characterized by the following steps:

   (a) selecting from the search key related to each dimension a predetermined dimension-specific number (k_i) of bits and forming therefrom a search word on the basis of which the address of the next node is sought from the internal node at the root level of the tree-shaped hierarchy, and proceeding to said node,
   (b) selecting from the unselected bits in the search key related to each dimension a predetermined dimension-specific number (k_i) of bits and forming therefrom a search word with which the address of a further new node at a lower level is sought from the array of the node that has been accessed,
   (c) repeating step (b) until an empty element has been encountered or until the address of the new node at a lower level is the address of a leaf node,
   (d) storing said data unit in the leaf node or alternatively storing a pointer in the leaf node and said data unit at the storage location indicated by the pointer.

2. A method as claimed in claim 1, **characterized in that** a new leaf node wherein storage is performed is created after step (c), if necessary.

3. A method as claimed in claim 1, **characterized in that** said dimension-specific number (k_i) is selected in such a way that the structure has equal depth in the direction of all dimensions and hence will favour partial key retrievals equitably in all directions.

4. A method as claimed in claim 1, **characterized in that** said dimension-specific number (k_i) is selected in such a way that the structure has greater depth in the direction of a given dimension/ given dimensions and hence will favour partial key retrievals **in that** direction/those directions.

5. A method as claimed in claim 1, **characterized in that** the search word is formed by setting the bits selected from the search key of each dimension in succession and interpreting the bit string thus obtained as a number indicating in the array of the internal node the element wherefrom the address of a lower node is sought.

6. A method for retrieving data identifiable by a search key from memory, the data being stored as data units in a dedicated storage space assigned for each data unit, wherein retrieval is performed by proceeding on the basis of a search key in a directory structure comprising a tree-shaped hierarchy having nodes at several different levels, wherein an individual node can be

   - an internal node comprising a multidimensional array wherein an individual element may contain the address of a lower node in the tree-shaped hierarchy and wherein an individual element may also be empty, or
   - a leaf node in which at least one data unit is stored or alternatively which contains at least one pointer to a stored data unit, **characterized by** the following steps:

   (a) selecting from the search key related to each dimension a predetermined dimension-specific number (k_i) of bits and forming therefrom a search word on the basis of which the address of the next node is sought from the internal node at the root level of the tree-shaped hierarchy, and proceeding to said node,
   (b) selecting from the unselected bits in the search key related to each dimension a predetermined dimension-specific number (k_i) of bits and forming therefrom a search word with which the address of a further new node at a lower level is sought from the array of the node that has been accessed,
   (c) repeating step (b) until an empty element has been encountered, until the address of the new node at a lower level is the address of a leaf node or until all bits of the search key have been selected, and
   if an empty element was not encountered in step (c),
   (d) reading from the leaf node one or more data units contained therein or alternatively reading from the

leaf node the pointer contained therein and reading the data unit at the location indicated by the pointer.

7. A method as claimed in claim 6, **characterized** in that said dimension-specific number (k_i) is selected in such a way that the structure has equal depth in the direction of all dimensions and hence will favour partial key retrievals equitably in all directions.

8. A method as claimed in claim 6, **characterized** in that said dimension-specific number (k_i) is selected in such a way that the structure has greater depth in the direction of a given dimension/ given dimensions and hence will favour partial key retrieval in that direction/those directions.

9. A method as claimed in claim 6, **characterized** in that the search word is formed by setting the bits selected from the search key of each dimension in succession and interpreting the bit string thus obtained as a number indicating in the array of the internal node the element wherefrom the address of a lower node is sought.

10. A method for storing data identifiable by a search key in memory, the data being stored as data units in a dedicated storage space assigned for each data unit, wherein storage is performed by proceeding, on the basis of a search key associated with the data unit to be stored, in a directory structure having a single node comprising a multidimensional array wherein an individual element may contain a data unit or a pointer to a data unit or wherein an individual element may also be empty,
characterized by
selecting from the search key related to each dimension a predetermined dimension-specific number (k_i) of bits and forming therefrom a search word on the basis of which an element is sought from the array of the node, and if said element is empty, storing the data unit in said element or alternatively storing a pointer in the element and said data unit at the storage location indicated by the pointer.

11. A method as claimed in claim 10, **characterized** in that the search word is formed by setting the bits selected from the search key of each dimension in succession and interpreting the bit string thus obtained as a number indicating said element.

12. A method for retrieving data identifiable by a search key from memory, the data being stored as data units in a dedicated storage space assigned for each data unit, wherein retrieval is performed by proceeding, on the basis of a search key associated with the data unit to be stored, in a directory structure having a single node comprising a multidimensional array wherein an individual node may contain a data unit or a pointer to said data unit or wherein an individual node may also be empty,
characterized by
selecting from the search key related to each dimension a predetermined dimension-specific (k_i) number of bits and forming therefrom a search word on the basis of which an element is sought from the array of the node, and if said element is empty, storing the data unit in said element or alternatively storing a pointer in said element and the data unit at the storage location indicated by the pointer.

13. A method as claimed in claim 12, **characterized** in that the search word is formed by setting the bits selected from the search key of each dimension in succession and interpreting the bit string thus obtained as a number indicating said element.

14. An associative memory arrangement for storing data in electrical form, the data being stored as data units in a dedicated storage space assigned for each data unit in the memory, the memory arrangement comprising a directory structure which is a multidimensional digital trie structure comprising a tree-shaped hierarchy having nodes at several different levels, wherein a node can be

- an internal node comprising a multidimensional array wherein an individual non-empty element comprises a pointer (31) guiding progress in the directory structure, or
- a leaf node containing at least one data unit or one or more pointers to a data unit stored in the memory,

each data unit to be stored in the memory being associated with a multidimensional search key comprising a search key for each dimension, on the basis of which search keys progress is made in the directory structure, the directory structure returning the data unit associated with the multidimensional search key employed in each case, **characterized in that** the size of an individual multidimensional internal node in the directory structure has been fixed by predetermining the number of search key bits k_i to be searched in each dimension independently of

the other dimensions, the size of the internal node in each dimension being 2^{k_i}.

Patentansprüche

1. Verfahren zur Speicherung von durch einen Suchcode identifizierbaren Daten in einem Speicher, die als Dateneinheiten in einem für jede Dateneinheit zugeordneten dedizierten Speicherraum gespeichert werden, wobei die Speicherung durch Fortschreiten in einer Verzeichnisstruktur mit einer baumförmigen Hierarchie mit Knoten an mehreren verschiedenen Stufen auf der Grundlage eines mit der zu speichernden Dateneinheit verbundenen Suchcodes durchgeführt wird, wobei ein einzelner Knoten

   - ein innerer Knoten mit einem mehrdimensionalen Array, in dem ein einzelnes Element die Adresse eines unteren Knotens in der baumförmigen Hierarchie enthalten kann, und in dem ein einzelnes Element auch leer sein kann, oder
   - ein Blattknoten, in dem zumindest eine Dateneinheit gespeichert ist, oder der zumindest einen Zeiger auf eine gespeicherte Dateneinheit enthält,

   sein kann, **gekennzeichnet durch** die folgenden Schritte:

   (a) Auswählen einer vorbestimmten dimensionsspezifischen Anzahl (k_i) an Bits aus dem auf jede Dimension bezogenen Suchcode und Ausbilden eines Suchwortes daraus, auf dessen Grundlage die Adresse des nächsten Knotens von dem inneren Knoten aus an der Wurzelstufe der baumförmigen Hierarchie gesucht wird, und Fortschreiten zu dem Knoten,
   (b) Auswählen einer vorbestimmten dimensionsspezifischen Anzahl (k_i) an Bits aus den nichtausgewählten Bits in dem auf jede Dimension bezogenen Suchcode und Ausbilden eines Suchworte daraus, mit dem die Adresse eines weiteren neuen Knotens an einer unteren Stufe von dem Array des Knotens aus gesucht wird, auf den zugegriffen wurde,
   (c) Wiederholen des Schritts (b), bis auf ein leeres Element getroffen wurde oder bis die Adresse des neuen Knotens an einer unteren Stufe die Adresse eines Blattknotens ist,
   (d) Speichern der Dateneinheit in dem Blattknoten oder, alternativ dazu, Speichern eines Zeigers in dem Blattknoten und der Dateneinheit am **durch** den Zeiger angezeigten Speicherort.

2. Verfahren nach Anspruch 1, **dadurch gekennzeichnet, dass** ein neuer Blattknoten, in dem die Speicherung durchgeführt wird, bei Bedarf nach Schritt (c) erzeugt wird.

3. Verfahren nach Anspruch 1, **dadurch gekennzeichnet, dass** die dimensionsspezifische Anzahl (k_i) derart ausgewählt wird, dass die Struktur die gleiche Tiefe in der Richtung aller Dimensionen hat, und somit Teilcodewiedergewinnungen gerecht in allen Richtungen begünstigt.

4. Verfahren nach Anspruch 1, **dadurch gekennzeichnet, dass** die dimensionsspezifische Anzahl (k_i) derart ausgewählt wird, dass die Struktur eine größere Tiefe in der Richtung einer gegebenen Dimension / gegebener Dimensionen hat, und somit Teilcodewiedergewinnungen in dieser Richtung / diesen Richtungen begünstigt.

5. Verfahren nach Anspruch 1, **dadurch gekennzeichnet, dass** das Suchwort durch Hintereinandersetzen der aus dem Suchcode jeder Dimension ausgewählten Bits und Interpretieren der so erhaltenen Bitkette als Zahl gebildet wird, die in dem Array des inneren Knotens das Element bezeichnet, von dem aus die Adresse eines unteren Knotens gesucht wird.

6. Verfahren zur Wiedergewinnung von durch einen Suchcode identifizierbaren Daten aus einem Speicher, die als Dateneinheiten in einem für jede Dateneinheit zugeordneten dedizierten Speicherraum gespeichert werden, wobei die Wiedergewinnung durch Fortschreiten in einer Verzeichnisstruktur mit einer baumförmigen Hierarchie mit Knoten an mehreren verschiedenen Stufen auf der Grundlage eines Suchcodes durchgeführt wird, wobei ein einzelner Knoten

   - ein innerer Knoten mit einem mehrdimensionalen Array, in dem ein einzelnes Element die Adresse eines unteren Knotens in der baumförmigen Hierarchie enthalten kann, und in dem ein einzelnes Element auch leer sein kann, oder
   - ein Blattknoten, in dem zumindest eine Dateneinheit gespeichert ist, oder der alternativ dazu zumindest einen

Zeiger auf eine gespeicherte Dateneinheit enthält,

sein kann, **gekennzeichnet durch** die folgenden Schritte:

(a) Auswählen einer vorbestimmten dimensionsspezifischen Anzahl (k_i) an Bits aus dem auf jede Dimension bezogenen Suchcode und Ausbilden eines Suchwortes daraus, auf dessen Grundlage die Adresse des nächsten Knotens von dem inneren Knoten aus an der Wurzelstufe der baumförmigen Hierarchie gesucht wird, und Fortschreiten zu dem Knoten,
(b) Auswählen einer vorbestimmten dimensionsspezifischen Anzahl (k_i) an Bits aus den nichtausgewählten Bits in dem auf jede Dimension bezogenen Suchcode und Ausbilden eines Suchworte daraus, mit dem die Adresse eines weiteren neuen Knotens an einer unteren Stufe von dem Array des Knotens aus gesucht wird, auf den zugegriffen wurde,
(c) Wiederholen des Schritts (b), bis auf ein leeres Element getroffen wurde, bis die Adresse des neuen Knotens an einer unteren Stufe die Adresse eines Blattknotens ist, oder bis alle Bits des Suchcodes ausgewählt wurden, und

wenn in Schritt (c) auf kein leeres Element getroffen wurde,

(d) Lesen einer oder mehrerer darin enthaltener Dateneinheiten aus dem Blattknoten, oder, alternativ dazu, Lesen des darin enthaltenen Zeigers aus dem Blattknoten und Lesen der Dateneinheit am **durch** den Zeiger angezeigten Ort.

7. Verfahren nach Anspruch 6, **dadurch gekennzeichnet, dass** die dimensionsspezifische Anzahl (k_i) derart ausgewählt wird, dass die Struktur die gleiche Tiefe in der Richtung aller Dimensionen hat, und somit Teilcodewiedergewinnungen gerecht in allen Richtungen begünstigt.

8. Verfahren nach Anspruch 6, **dadurch gekennzeichnet, dass** die dimensionsspezifische Anzahl (k_i) derart ausgewählt wird, dass die Struktur eine größere Tiefe in der Richtung einer gegebenen Dimension / gegebener Dimensionen hat, und somit Teilcodewiedergewinnungen in dieser Richtung / diesen Richtungen begünstigt.

9. Verfahren nach Anspruch 6, **dadurch gekennzeichnet, dass** das Suchwort durch Hintereinandersetzen der aus dem Suchcode jeder Dimension ausgewählten Bits und Interpretieren der so erhaltenen Bitkette als Zahl gebildet wird, die in dem Array des inneren Knotens das Element bezeichnet, von dem aus die Adresse eines unteren Knotens gesucht wird.

10. Verfahren zur Speicherung von durch einen Suchcode identifizierbaren Daten in einem Speicher, die als Dateneinheiten in einem für jede Dateneinheit zugeordneten dedizierten Speicherraum gespeichert werden, wobei die Speicherung durch Fortschreiten in einer Verzeichnisstruktur mit einem einzigen Knoten mit einem mehrdimensionalen Array auf der Grundlage eines mit der zu speichernden Dateneinheit verbundenen Suchcodes durchgeführt wird, wobei in dem Array ein einzelnes Element eine Dateneinheit oder einen Zeiger auf eine Dateneinheit enthalten kann oder ein einzelnes Element auch leer sein kann,
gekennzeichnet durch die Schritte:

Auswählen einer vorbestimmten dimensionsspezifischen Anzahl (k_i) an Bits aus dem auf jede Dimension bezogenen Suchcode und Ausbilden eines Suchwortes daraus, auf dessen Grundlage ein Element aus dem Array des Knotens gesucht wird, und ist das Element leer, Speichern der Dateneinheit in dem Element oder, alternativ dazu, Speichern eines Zeigers in dem Element und der Dateneinheit am **durch** den Zeiger angezeigten Speicherort.

11. Verfahren nach Anspruch 10, **dadurch gekennzeichnet, dass** das Suchwort durch Hintereinandersetzen der aus dem Suchcode jeder Dimension ausgewählten Bits und Interpretieren der so erhaltenen Bitkette als Zahl gebildet wird, die das Element anzeigt.

12. Verfahren zur Wiedergewinnung von durch einen Suchcode identifizierbaren Daten aus einem Speicher, die als Dateneinheiten in einem für jede Dateneinheit zugeordneten dedizierten Speicherraum gespeichert werden, wobei die Wiedergewinnung durch Fortschreiten in einer Verzeichnisstruktur mit einem einzigen Knoten mit einem mehrdimensionalen Array auf der Grundlage eines mit der zu speichernden Dateneinheit verbundenen Suchcodes durchgeführt wird, wobei in dem Array ein einzelnes Element eine Dateneinheit oder einen Zeiger auf die Dateneinheit enthalten kann oder ein einzelnes Element auch leer sein kann,
gekennzeichnet durch die Schritte:

Auswählen einer vorbestimmten dimensionsspezifischen Anzahl (k_i) an Bits aus dem auf jede Dimension

bezogenen Suchcode und Ausbilden eines Suchwortes daraus, auf dessen Grundlage ein Element aus dem Array des Knotens gesucht wird, und ist das Element leer, Speichern der Dateneinheit in dem Element oder, alternativ dazu, Speichern eines Zeigers in dem Element und der Dateneinheit am **durch** den Zeiger angezeigten Speicherort.

13. Verfahren nach Anspruch 12, **dadurch gekennzeichnet, dass** das Suchwort durch Hintereinandersetzen der aus dem Suchcode jeder Dimension ausgewählten Bits und Interpretieren der so erhaltenen Bitkette als Zahl gebildet wird, die das Element anzeigt.

14. Assoziative Speicheranordnung zur Speicherung von Daten in elektronischer Form, die als Dateneinheiten in einem für jede Dateneinheit zugeordneten dedizierten Speicherraum in dem Speicher gespeichert werden, wobei die Speicheranordnung eine Verzeichnisstruktur hat, die eine mehrdimensionale digitale Baumstruktur mit einer baumförmigen Hierarchie mit Knoten an mehreren verschiedenen Stufen ist, wobei ein Knoten

- ein innerer Knoten mit einem mehrdimensionalen Array, in dem ein einzelnes nicht leeres Element einen Zeiger (31) umfasst, der das Fortschreiten in der Verzeichnisstruktur steuert, oder
- ein Blattknoten mit zumindest einer Dateneinheit oder einem oder mehreren Zeigern auf eine in dem Speicher gespeicherte Dateneinheit

sein kann, wobei jede in dem Speicher zu speichernde Dateneinheit mit einem mehrdimensionalen Suchcode mit einem Suchcode für jede Dimension verbunden ist, wobei auf der Grundlage dieser Suchcodes ein Fortschreiten in der Verzeichnisstruktur durchgeführt wird, die die Dateneinheit zurückgibt, die mit dem im jeweiligen Fall angewendeten mehrdimensionalen Suchcode verbunden ist,
dadurch gekennzeichnet, dass
die Größe eines einzelnen mehrdimensionalen inneren Knotens in der Verzeichnisstruktur durch Vorbestimmung der Anzahl an zu suchenden Suchcodebits k_i in jeder Dimension unabhängig von den anderen Dimensionen festgelegt wurde, wobei die Größe des inneren Knotens in jeder Dimension 2^{ki} beträgt.

Revendications

1. Procédé destiné à stocker dans une mémoire des données identifiables par une clé de recherche, les données étant stockées sous forme d'unités de données dans un espace de stockage dédié, attribué à chaque unité de données, ledit stockage étant effectué, en fonction d'une clé de recherche associée à l'unité de données à stocker, suivant une structure en répertoires à hiérarchie arborescente, comprenant des noeuds à plusieurs niveaux différents, chaque noeud individuel pouvant être

   - un noeud interne comprenant un tableau à plusieurs dimensions dans lequel un élément individuel peut contenir l'adresse d'un noeud de niveau inférieur dans la hiérarchie arborescente et dans lequel un élément individuel peut également être vide, ou
   - un noeud feuille dans lequel au moins une unité de données est stockée ou qui contient au moins un pointeur correspondant à une unité de données stockée, **caractérisé par** les étapes qui consistent à :

     (a) choisir, à partir de la clé de recherche associée à chaque dimension, un nombre de bits prédéterminé (k_i), spécifique de la dimension, et former, à partir de ce nombre, un mot de recherche en fonction duquel l'adresse du noeud suivant est recherchée à partir du noeud interne au niveau racine de la hiérarchie arborescente, et passer audit noeud,
     (b) choisir, à partir des bits non sélectionnés dans la clé de recherche associée à chaque dimension, un nombre de bits prédéterminé (k;), spécifique de la dimension, et former, à partir de ce nombre, un mot de recherche avec lequel l'adresse d'encore un nouveau noeud, à un niveau inférieur, est recherchée dans le tableau du noeud qui a été accédé,
     (c) répéter l'étape (b) jusqu'à ce qu'un élément vide soit rencontré ou jusqu'à ce que l'adresse du nouveau noeud à un niveau inférieur soit l'adresse d'un noeud feuille,
     (d) stocker ladite unité de données dans le noeud feuille ou, d'une autre manière, stocker un pointeur dans le noeud feuille et ladite unité de données à l'endroit de stockage indiqué par le pointeur.

2. Procédé selon la revendication 1,
   caractérisé en ce que, si nécessaire, un nouveau noeud feuille dans lequel effectuer un stockage est créé

après l'étape (c).

3. Procédé selon la revendication 1,
caractérisé en ce que ledit nombre (k_i) spécifique de la dimension est choisi de telle sorte que la structure ait une profondeur égale dans la direction de toutes les dimensions, et favorise donc des récupérations de clés partielles qui soient équitables dans toutes les directions.

4. Procédé selon la revendication 1,
caractérisé en ce que ledit nombre (k_i) spécifique de la dimension est choisi de telle sorte que la structure ait une plus grande profondeur dans la direction d'une ou plusieurs dimensions données, et favorise donc des récupérations de clés partielles dans cette ou ces directions.

5. Procédé selon la revendication 1,
caractérisé en ce que le mot de recherche est formé par le fait de fixer les uns à la suite des autres les bits sélectionnés à partir de la clé de recherche de chaque dimension, et interpréter la chaîne de bits ainsi obtenue comme un nombre indiquant, dans le tableau du noeud interne, l'élément à partir duquel l'adresse d'un noeud de niveau inférieur est recherchée.

6. Procédé destiné à récupérer dans une mémoire des données identifiables par une clé de recherche, lesdites données étant stockées sous forme d'unités de données, dans un espace de stockage dédié, attribué à chaque unité de données, ladite récupération étant effectuée, en fonction d'une clé de recherche associée à l'unité de données à stocker, suivant une structure en répertoires à hiérarchie arborescente, comprenant des noeuds à plusieurs niveaux différents, chaque noeud individuel pouvant être

- un noeud interne comprenant un tableau à plusieurs dimensions dans lequel un élément individuel peut contenir l'adresse d'un noeud de niveau inférieur dans la hiérarchie arborescente et dans lequel un élément individuel peut également être vide, ou
- un noeud feuille dans lequel au moins une unité de données est stockée ou qui contient au moins un pointeur correspondant à une unité de données stockée, **caractérisé par** les étapes qui consistent à :

(a) choisir, à partir de la clé de recherche associée à chaque dimension, un nombre de bits prédéterminé (k_i), spécifique de la dimension, et former, à partir de ce nombre, un mot de recherche en fonction duquel l'adresse du noeud suivant est recherchée à partir du noeud interne au niveau racine de la hiérarchie arborescente, et passer audit noeud,
(b) choisir, à partir des bits non sélectionnés dans la clé de recherche associée à chaque dimension, un nombre de bits prédéterminé (k_i), spécifique de la dimension, et former, à partir de ce nombre, un mot de recherche avec lequel l'adresse d'encore un nouveau noeud, à un niveau inférieur, est recherchée dans le tableau du noeud qui a été accédé,
(c) répéter l'étape (b) jusqu'à ce qu'un élément vide soit rencontré ou jusqu'à ce que l'adresse du nouveau noeud à un niveau inférieur soit l'adresse d'un noeud feuille, ou bien encore, jusqu'à ce que tous les bits de la clé de recherche aient été sélectionnés, et
si un élément vide n'a pas été rencontré dans l'étape (c),
(d) lire une ou plusieurs unités de données qui sont contenues dans le noeud feuille, ou, d'une autre manière, lire le pointeur qui est contenu dans le noeud feuille et lire l'unité de données à l'endroit indiqué par le pointeur.

7. Procédé selon la revendication 6,
caractérisé en ce que ledit nombre (k_i) spécifique de la dimension est choisi de telle sorte que la structure ait une profondeur égale dans la direction de toutes les dimensions, et favorise donc des récupérations de clés partielles qui soient équitables dans toutes les directions.

8. Procédé selon la revendication 6,
caractérisé en ce que ledit nombre (k_i) spécifique de la dimension est choisi de telle sorte que la structure ait une plus grande profondeur dans la direction d'une ou plusieurs dimensions données, et favorise donc des récupérations de clés partielles dans cette ou ces directions.

9. Procédé selon la revendication 6,
caractérisé en ce que le mot de recherche est formé par le fait de fixer les uns à la suite des autres les bits

sélectionnés à partir de la clé de recherche de chaque dimension, et interpréter la chaîne de bits ainsi obtenue comme un nombre indiquant, dans le tableau du noeud interne, l'élément à partir duquel l'adresse d'un noeud de niveau inférieur est recherchée.

10. Procédé destiné à stocker dans une mémoire des données identifiables par une clé de recherche, les données étant stockées sous forme d'unités de données dans un espace de stockage dédié, attribué à chaque unité de données, ledit stockage étant effectué, en fonction d'une clé de recherche associée à l'unité de données à stocker, suivant une structure en répertoires ayant un seul noeud, comprenant un tableau à plusieurs dimensions dans lequel un élément individuel peut contenir une unité de données ou un pointeur correspondant à une unité de données, ou dans lequel un élément individuel peut également être vide, **caractérisé par** les étapes qui consistent à :

choisir, à partir de la clé de recherche associée à chaque dimension, un nombre de bits prédéterminé (k_i), spécifique de la dimension, et former, à partir de ce nombre, un mot de recherche en fonction duquel un élément est recherché à partir du tableau du noeud, et, si ledit élément est vide, stocker l'unité de données dans ledit élément ou, d'une autre manière, stocker un pointeur dans l'élément et ladite unité de données à l'endroit de stockage indiqué par le pointeur.

11. Procédé selon la revendication 10,

caractérisé en ce que le mot de recherche est formé par le fait de fixer les uns à la suite des autres les bits sélectionnés à partir de la clé de recherche de chaque dimension, et interpréter la chaîne de bits ainsi obtenue comme un nombre indiquant ledit élément.

12. Procédé destiné à récupérer dans une mémoire des données identifiables par une clé de recherche, lesdites données étant stockées sous forme d'unités de données, dans un espace de stockage dédié, attribué à chaque unité de données, ladite récupération étant effectuée, en fonction d'une clé de recherche associée à l'unité de données à stocker, suivant une structure en répertoires ayant un seul noeud, comprenant un tableau à plusieurs dimensions dans lequel un élément individuel peut contenir une unité de données ou un pointeur correspondant à une unité de données, ou dans lequel dans lequel un élément individuel peut également être vide, **caractérisé par** les étapes qui consistent à :

choisir, à partir de la clé de recherche associée à chaque dimension, un nombre de bits prédéterminé (k_i), spécifique de la dimension, et former, à partir de ce nombre, un mot de recherche en fonction duquel un élément est recherché à partir du tableau du noeud, et, si ledit élément est vide, stocker l'unité de données dans ledit élément ou, d'une autre manière, stocker un pointeur dans l'élément et ladite unité de données à l'endroit de stockage indiqué par le pointeur.

13. Procédé selon la revendication 12,

caractérisé en ce que le mot de recherche est formé par le fait de fixer les uns à la suite des autres les bits sélectionnés à partir de la clé de recherche de chaque dimension, et interpréter la chaîne de bits ainsi obtenue comme un nombre indiquant ledit élément.

14. Agencement de mémoire associative destiné à stocker des données sous forme électrique, les données étant stockées sous forme d'unités de données, dans un espace de stockage dédié, attribué à chaque unité de données dans la mémoire, ledit agencement de mémoire comprenant une structure en répertoires qui est une structure en arbre dérivé numérique, à plusieurs dimensions, ayant une hiérarchie arborescente comprenant des noeuds à plusieurs niveaux différents, un noeud pouvant être

- un noeud interne comprenant un tableau à plusieurs dimensions, dans lequel un élément non vide individuel comprend un pointeur (31) guidant la progression dans la structure en répertoires, ou
- un noeud feuille contenant au moins une unité de données ou bien un ou plusieurs pointeurs correspondant à une unité de données stockée dans la mémoire,

chacune unité de données à stocker dans la mémoire étant associée à une clé de recherche à plusieurs dimensions, comprenant une clé de recherche pour chaque dimension, en fonction desquelles clés de recherche la progression se fait suivant une structure en répertoires, la structure en répertoires renvoyant l'unité de données associée à la clé de recherche à plusieurs dimensions utilisée dans chaque cas, **caractérisé en ce que** la taille d'un noeud interne à plusieurs dimensions individuel de la structure en répertoire a été fixée par le fait de préétablir le nombre de bits de clé de recherche k_i à rechercher dans chaque dimension, indépendamment des autres dimensions, la taille du noeud interne dans chaque dimension étant égale à 2^{k_i}.

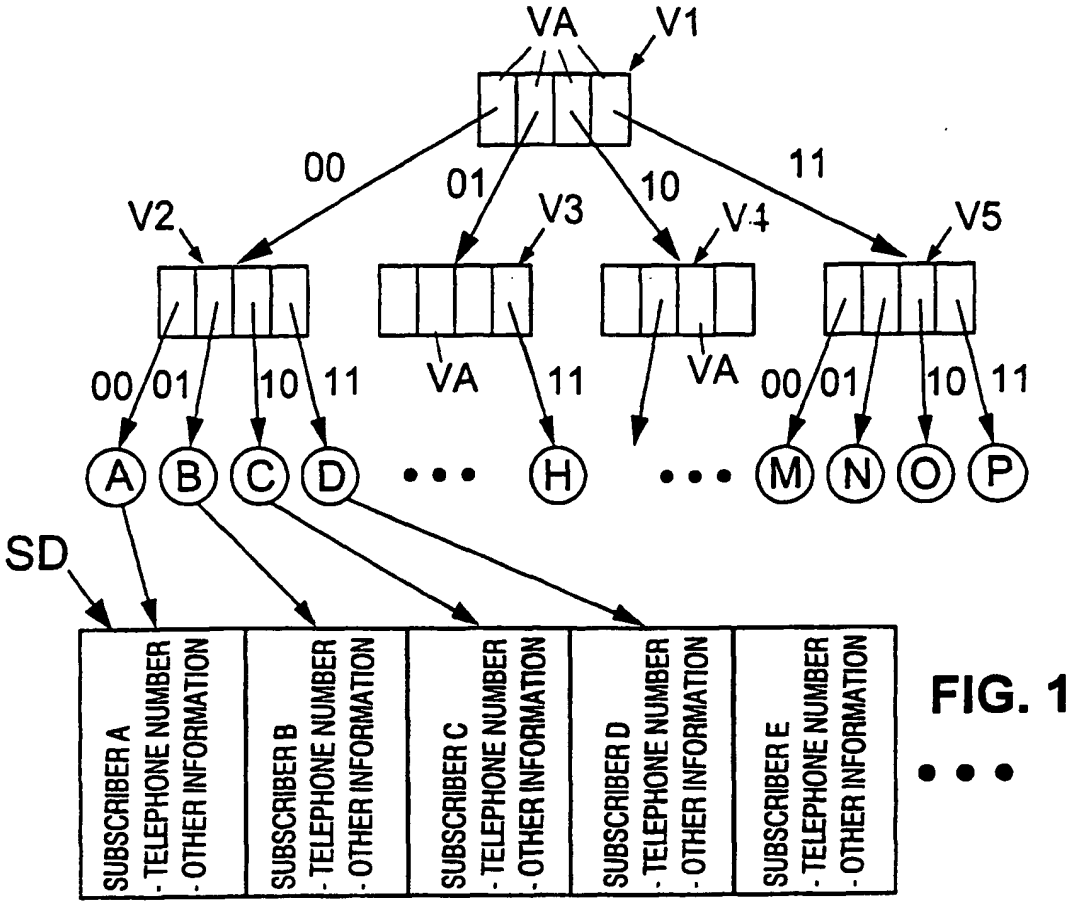

VA
V1
00
01
10
11
V2
V3
V4
V5
00
01
10
11
VA
11
VA
00
01
10
11
A
B
C
D
H
M
N
O
P
SD
SUBSCRIBER A
- TELEPHONE NUMBER
- OTHER INFORMATION
SUBSCRIBER B
- TELEPHONE NUMBER
- OTHER INFORMATION
SUBSCRIBER C
- TELEPHONE NUMBER
- OTHER INFORMATION
SUBSCRIBER D
- TELEPHONE NUMBER
- OTHER INFORMATION
SUBSCRIBER E
- TELEPHONE NUMBER
- OTHER INFORMATION

FIG. 1

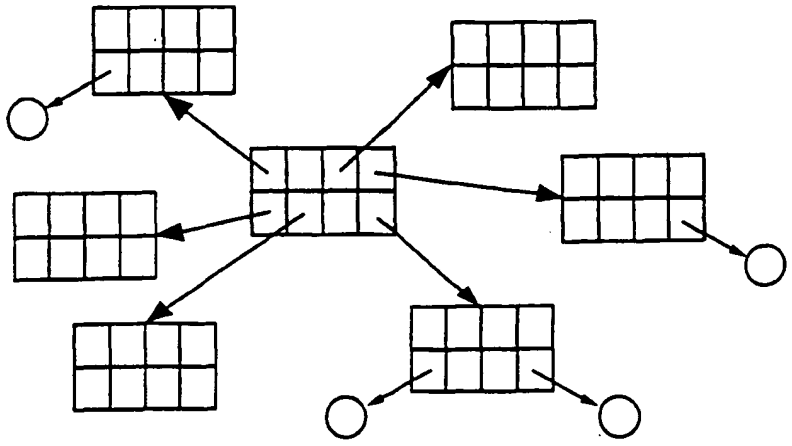


FIG. 2

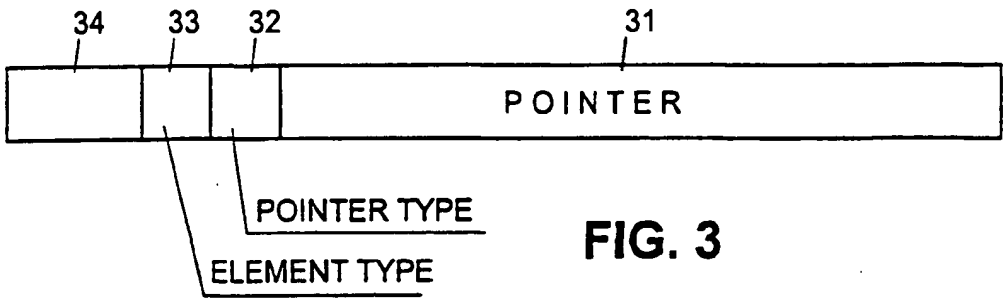

34
33
32
31
POINTER
POINTER TYPE
ELEMENT TYPE

FIG. 3

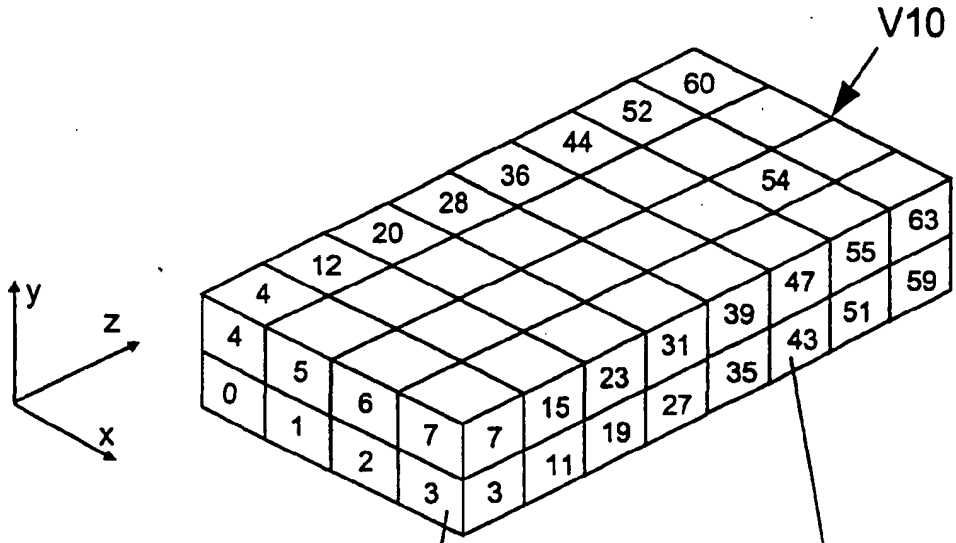
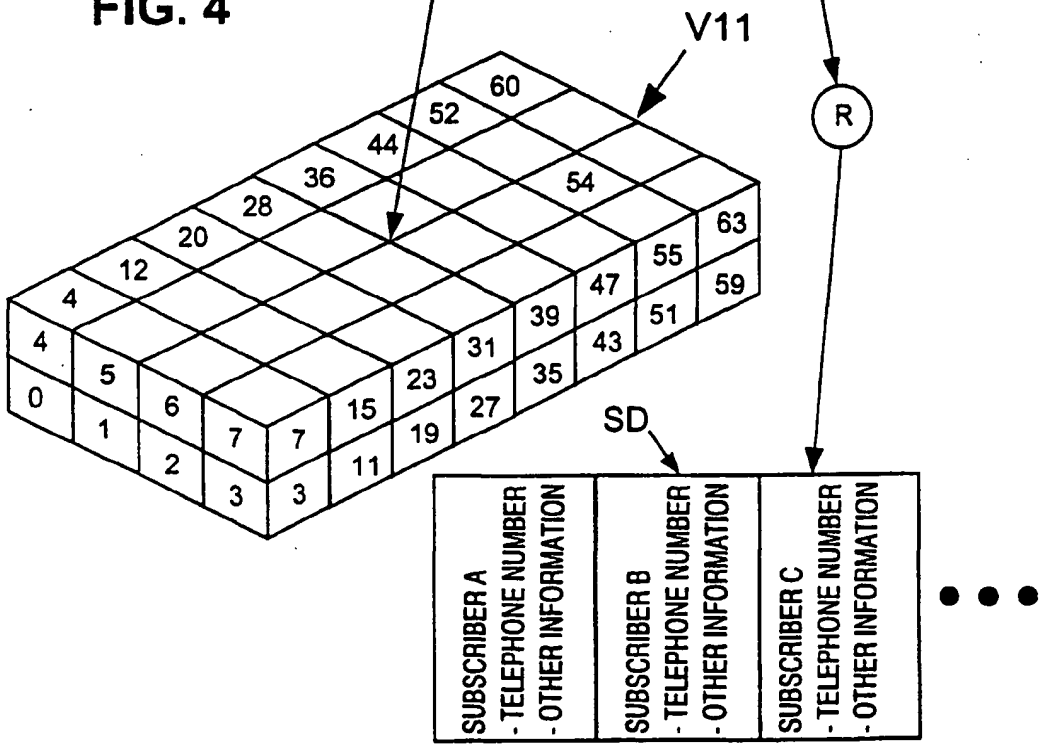

V10
y
z
x
0
1
2
3
4
5
6
7
7
3
4
11
12
15
19
20
23
27
28
31
35
36
39
43
44
47
51
52
54
55
59
60
63

FIG. 4


V11
R
SD
SUBSCRIBER A
- TELEPHONE NUMBER
- OTHER INFORMATION
SUBSCRIBER B
- TELEPHONE NUMBER
- OTHER INFORMATION
SUBSCRIBER C
- TELEPHONE NUMBER
- OTHER INFORMATION

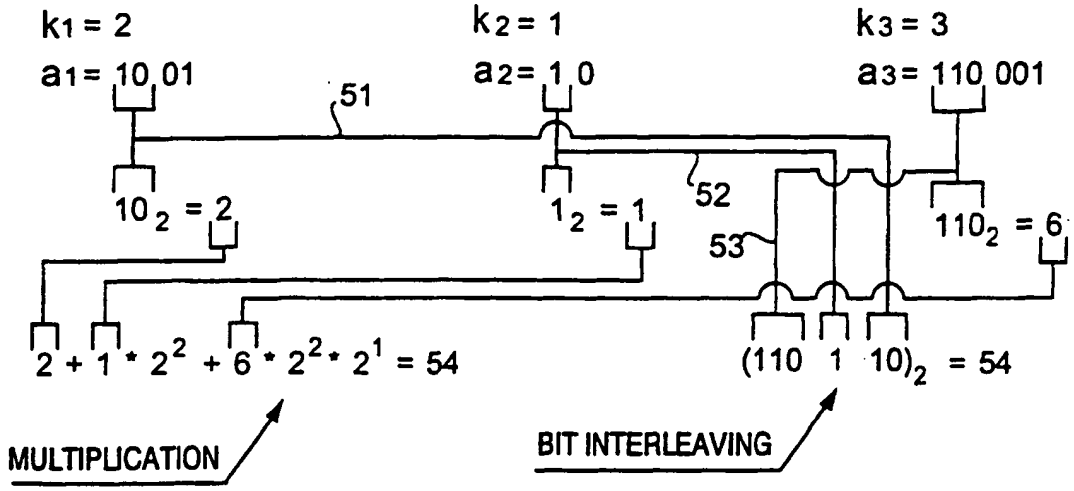

k1 = 2
a1 = 10 01
k2 = 1
a2 = 1 0
k3 = 3
a3 = 110 001
51
52
53
10 2 = 2
1 2 = 1
110 2 = 6
2 + 1 * 2^2 + 6 * 2^2 * 2^1 = 54
(110 1 10)2 = 54
MULTIPLICATION
BIT INTERLEAVING

FIG. 5

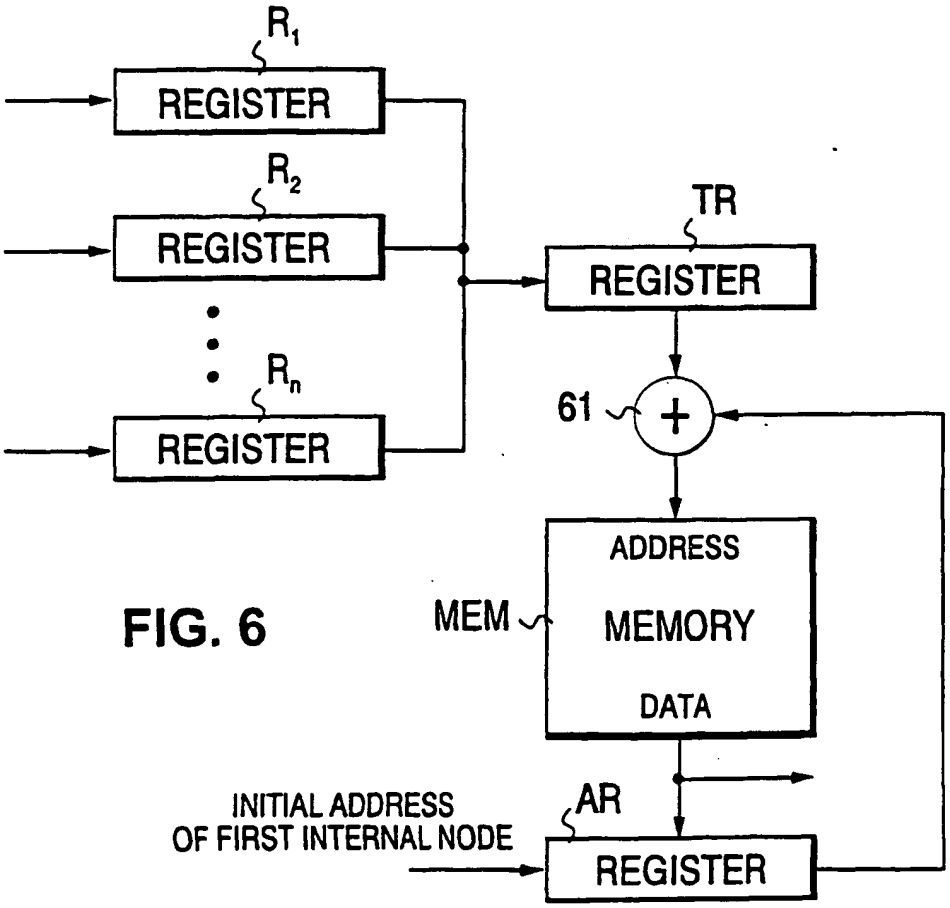

R1
REGISTER
R2
REGISTER
Rn
REGISTER
TR
REGISTER
61
+
ADDRESS
MEM
MEMORY
DATA
INITIAL ADDRESS
OF FIRST INTERNAL NODE
AR
REGISTER

FIG. 6